**以Web视图查看，这样图片可以看的清楚**

在我们用VS进行项目合作开发的过程中，SVN的提交控制是至关重要的，由于版本冲突造成的各种麻烦咱们已经遇到的够多了。所以，总结他们的经验教训，给我们也给其他人做个提醒。下面的第一部分是需要在正式开发之前需要做的，第二部分是开发的过程中需要注意的。

**一、排除不必要的提交**

**1.将编译性的文件排除在提交之外**

     由于编译性的文件（包括obj文件夹和bin文件夹）并不是源文件，它完全可以通过存储的源文件生成，一次提交的话会造成两方面的影响：1. 浪费服务器存储空间 2. 由于每个团队成员编译的结果可能并不一样，大大增加了版本冲突的几率。

**1.1 obj文件夹**

        obj目录是用来保存每个模块的编译结果，在.NET中，编译是分模块进行的，编译整个完成后会合并为一个.DLL或.EXE保存到bin目录下。因为每次编译时默认都是采用增量编译，即只重新编译改变了的模块，obj保存每个模块的编译结果，用来加快编译速度。是否采用增量编译，可以通过：项目属性—>配置属性—>高级—>增量编译来设置。

**1.2 bin文件夹**

        bin目录用来保存项目生成后程序集，后置代码类和其他非页面类编译后的DLL。它有Debug和Release两个版本，分别对应的文件夹为bin/Debug和bin/Release，这个文件夹是默认的输出路径，我们可以通过：项目属性—>配置属性—>输出路径来修改。

**2. 将属于每个用户的文件排除在提交之外**

**2.1 \*.csproj.user**

        .csproj.user文件是一个xml文件，用于存储当前项目的用户配置，可以使用记事本打开。例如：打开ASP.NET项目的.csproj.user文件后会看到一项：<StartAction>CurrentPage</StartAction>，它表示当按F5开始调试，或者Ctrl+F5开始运行时，首先打开的是VS的当前页。

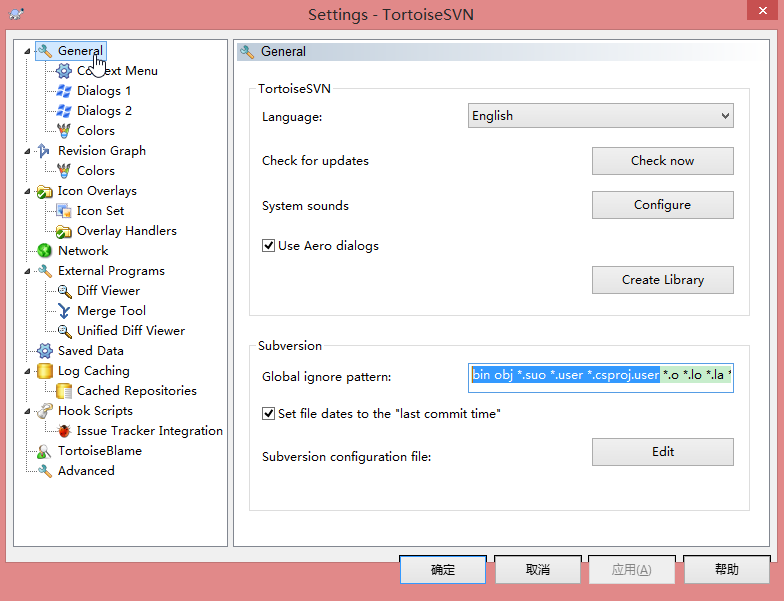
**2.2 \*.suo**

        \*.suo解决方案用户选项，记录所有将与解决方案建立关联的选项，以便在每次打开时，它都包含用户所做的自定义设置，比如VS布局以及项目最后编译的而又没有关掉的文件用于下次打开时用。删除之后，团队成员从服务器下载下来之后再次打开解决方案就会重新生成。

**3. 排除方法**

     在服务器上直接将以上的文件或者文件夹直接删除，同时要求各个团队成员自己在客户端将它们排除掉：

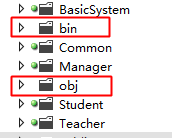
右击解决方案文件夹→TorToiseSVN→Settings→General，如下图：



        在“Subversion下的”“Globalignore pattern ”中添加要排除在提交之外的文件类型（以空格分隔）“bin obj \*.suo\*.user \*.csproj.user”即可。

        也可右击目标→TorToiseSVN→Unversion and add toignore list→文件类型，逐个排除。

        排除后以后再提交整个解决方案，被排除的文件类型都不会被提交：



   为减少冲突产生的几率，我们可以做双保险：在向服务器上传全新的解决方案之前将以上的5种类型全部删除，然后让团队成员再排除。以后大家在提交的时候按照下面的几个原则基本上就不会出什么问题了。

**二、提交的几个原则**

**1.先更新，再生成解决方案，最后提交。**

**1.1 先Update整个解决方案**

        团队成员可能会修改解决方案中的多个文件，所以更新的时候我们没有必要（也不建议这么做）去单个项目或者文件去更新，否则可能更新下来的代码由于只是部分导致生成的时候出现错误。

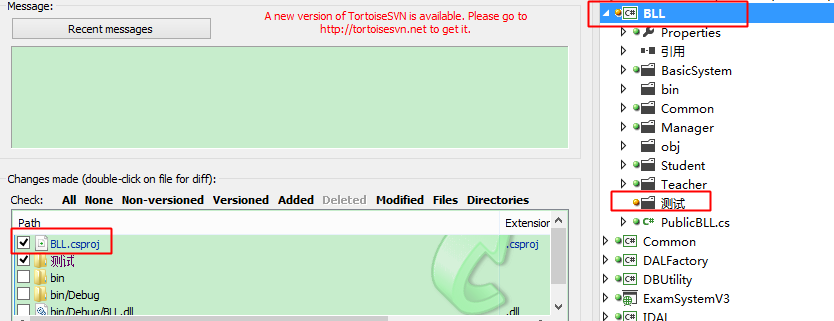
**1.2 然后保证在提交之前生成的解决方案没有错误**

        这个是提交之前最为重要的一步，一旦某个成员提交上了这种类型的代码，那么团队中的其他人都将无法进行调试。

**1.3 最后再提交，而且只Commit自己修改的类**

       只提交自己修改的类或者其他文件，避免因为误操作别人的类导致解决方案中的出错。其实可以通过SVN的权限控制各个成员负责的部分，可以精确到单个类（但这种方法也有不少局限性），这样就可以大大减少由于误操作引起的不必要的麻烦。

       对于新添加的类、文件或者文件夹等我们应该将他们所在的层进行提交（当然也可以对整个解决方案进行提交，但为了减少出错的几率采用此方法），提交的时候勾选上\*.csproj文件（默认是勾选的）和自己修改过的：



     这样就可以解决下载的解决方案中新建的这个文件夹未被包含在项目中的问题。

     而对于新添加的项目，则应该提交整个解决方案，方法跟上面类似。

**2.“微提交”**

      每实现一个小功能或者几段代码，生成没有错误之后就直接提交，也叫“保存提交”。这样可以通过SVN的版本控制，最大程度的减少因为后来的错误导致解决方案大范围修改情况的发生。

**3.未经组长同意，不得擅自使用“get lock”功能**

     就是对文件或者文件夹进行“锁定”的功能。只有对于特别重要的，属于只有自己能够修改的并且经过组长统一的才能够使用。否则其他人无法提交该文件或者文件夹。

**4.对SVN提交更新的信息采用明晰的标注(类似在代码里写的注释)**

     有了这个信息之后我们如果某个版本出现了错误就可以清晰的看到是因为谁修改了哪些部分导致的，很方便调试还原。如：我们规定的格式是：姓名—修改内容

**5.不要提交自己不明白的代码**

     代码在提交入SVN之后，你的代码将被项目成员所分享。如果提交了你不明白的代码，你看不懂，别人也看不懂，如果在以后出现了问题将会成为项目质量的隐患。因此在引入任何第三方代码之前，确保你对这个代码有一个很清晰的了解。

     注意：如果在之后引用了第3方dll，每次从服务器上全新下载的解决方案生成的时候肯定会出错，这时候就需要手动将dll添加到解决方案的相关bin文件夹里了，目前暂时没有想到一个比较好的解决方法。

      --解决方法之一：将需要引用第三方dll放到其他的文件夹公用即可，不要放到bin文件夹中了。

     以上几点做到之后，以后出现的问题只要是关于版本提交的基本上就能解决了。在实践的过程中可能还会有以上问题的出现，但只要我们能够清楚这些问题的来源，就能够比较快速、正确的处理掉。当然，还是需要大家去养成一个良好的提交习惯才能避免给大家带来麻烦。